Software Technology Support Center (STSC)

Helping Government Organizations Buy and Build Software Better

U.S. AIR FORCE

What is this System Really Suppose to do?

Or the "You Must be Kidding Syndrome."

Speaker

LESLIE (LES) DUPAIX
USAF STSC
les.dupaix@hill.af.mil

System analysis steps

1. Identify the customers need
2. Evaluate the proposed system for feasibility
3. Perform economic and technical analysis
4. Allocate function to hardware, software, people and data (database)
5. Establish cost and schedule constraints
6. Create a system definition

1. Identification of need

- **Meet with customer and end-user**
- **Understand who the "real" customers are (and uncover political agendas)**
- **Understand the requirements**
  - Know the difference between customer requirements and desirements
- **Establish goals**
  - Is current technology adequate to meet goals
  - What is potential market
  - How does system integrate with existing constraints
- **Document in a *System Concept Document***

2. Feasibility Study

- **Related to Risk Analysis and Risk management**
  - Development risks
  - Resources availability
  - Technology availability
  - Security

- **Evaluate feasibility of**
  - Economic feasibility
  - Technical feasibility
  - Legal feasibility
  - Examine other alternatives

3. Economic Analysis

- **Perform Cost/Benefit analysis**
- **Lots of factors to look at**
  - cost reduction (CR)
  - error reduction (ER)
  - increased flexibility or capability (IF)
  - increased speed (IS)
  - improvement in management planning and control (MC)
  - security

U.S. AIR FORCE

3. Technical analysis

- **Assessment of the technical viability of the system**
  - Is technology available?
  - What new materials, methods or processes are required?
- **Tool available**
  - Models and simulations
  - Probability theory
  - Queuing theory
  - Control theory

How to do technical analysis

- **Build a System Model**
  - Simple enough to understand, but as close to reality as possible (to yield valid results)
  - Highlight factors that are relevant or important, and (with discretion) suppress those not as important
  - Include relevant factors, and should give repeatable results
  - Small enough to be timely. If too big, consider breaking down into many smaller models
  - Make the model expandable and modifiable. This allows "tuning" of the model, and also expansion and inclusion of changing requirements

4. Allocate functions to...

- Hardware
- Software
- People
- Data (database)
- Other system elements
- Security

- Basically, an architectural model

5. Establish cost and schedule

- **Based on**
  - customer needs
  - economic feasibility
  - technical analysis
  - functional allocation
  - Security
- **Requires both management and customer buy-in**
- **Critical factor is typically time, not cost**

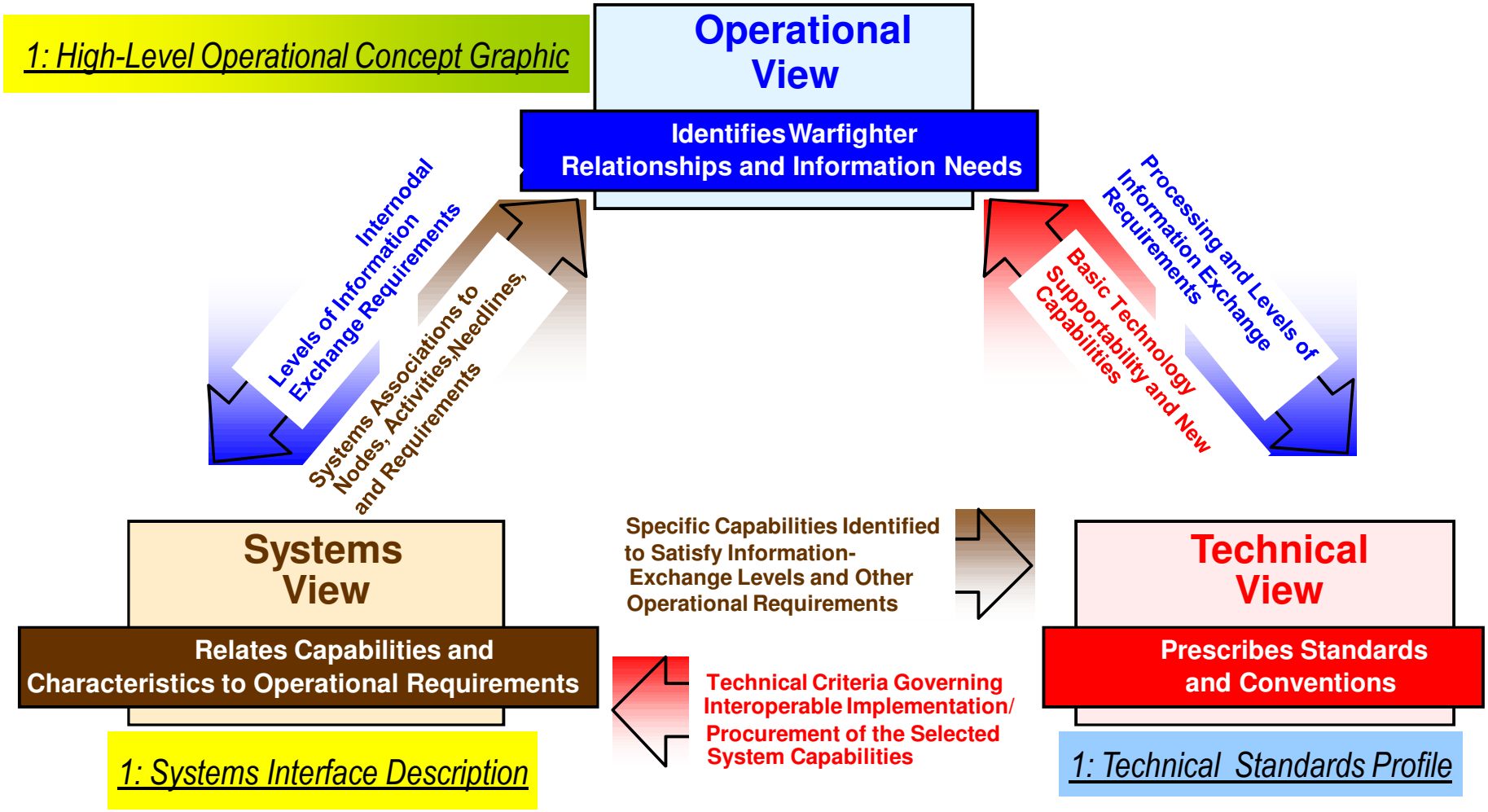
6. Create systems definition

- **The "blueprint" that guides the entire systems development.**
- **Explains what each area (hardware, software, etc) is responsible for.**
- **Explains interfaces between areas**
- **Forms the *Systems Specification,* which is the basis for future**
  - hardware engineering
  - software engineering
  - database engineering
  - human engineering
  - Security engineering

Modeling the system

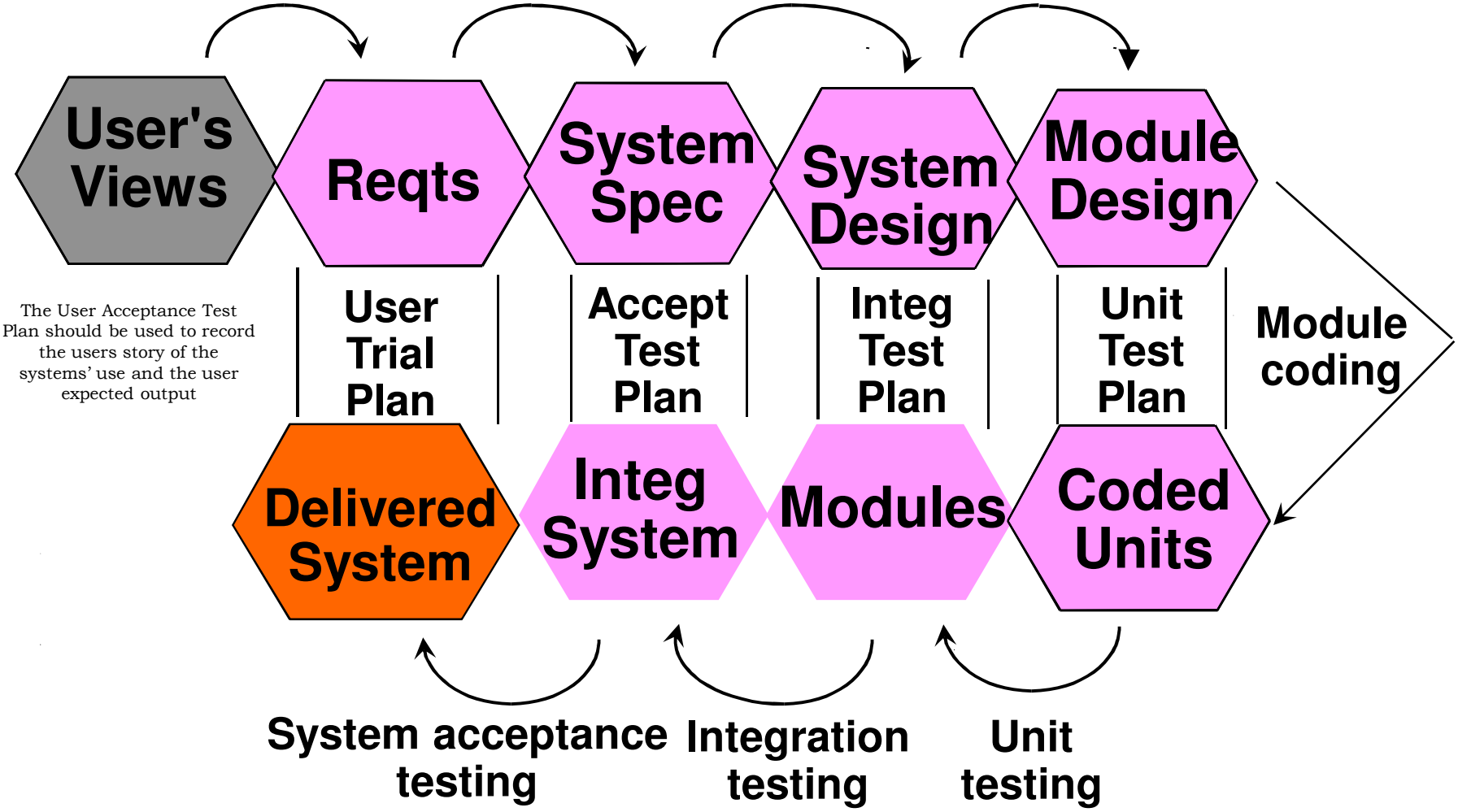
- It is necessary to define the boundaries of the system.
- Typically, a graphical representation or model is best for “first cut”
- Create different models or views of the system (operational view, system view, and technical view)
- DODAF is way to illustrate architecture
- An ACD (architecture context diagram) is another a high-level diagram that shows boundaries between the system and its’ environment. It lists external interfaces (informational boundaries)

Linkages Between the Views

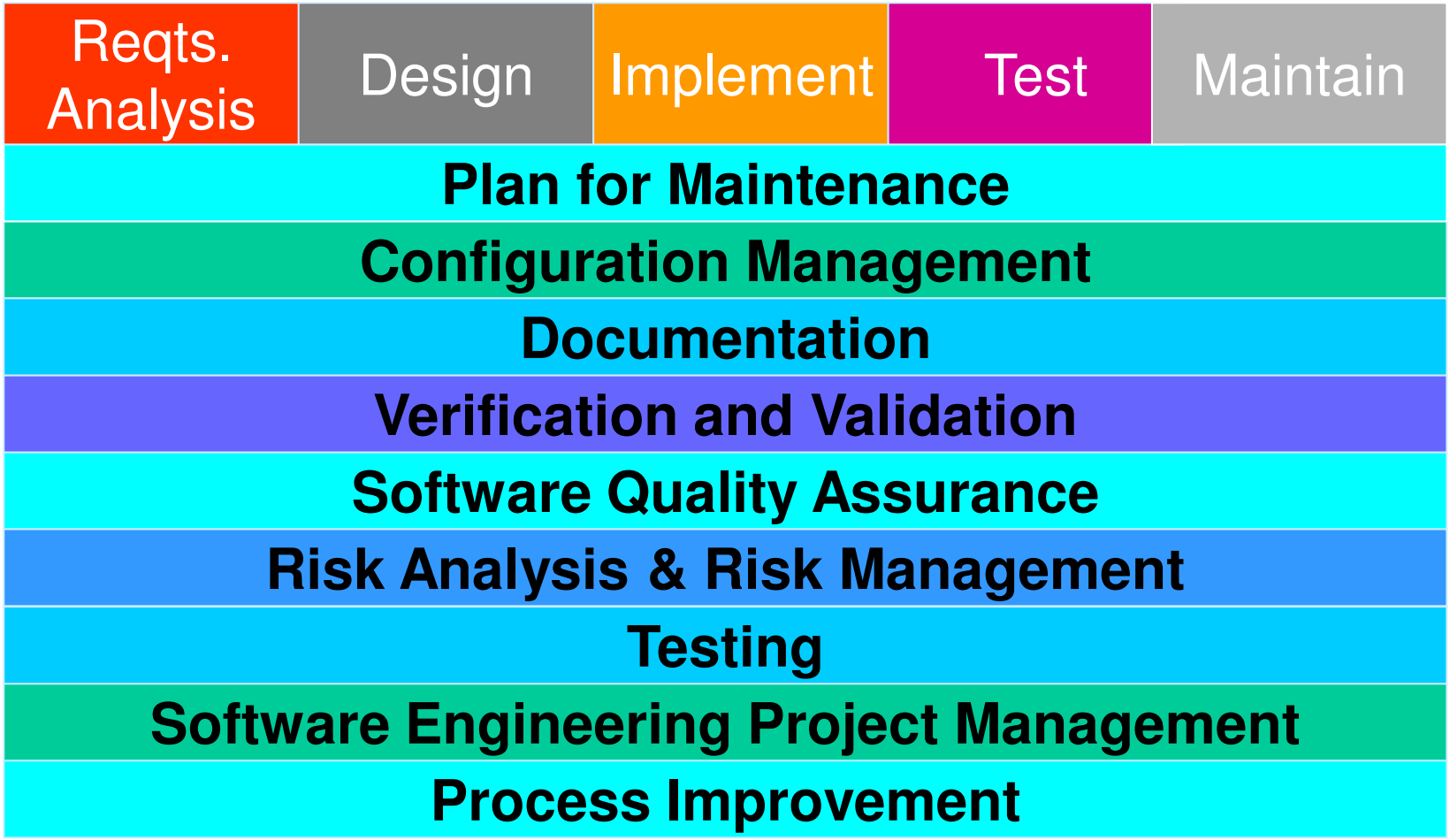


U.S. AIR FORCE

Lifecycle approach



Software Engineering

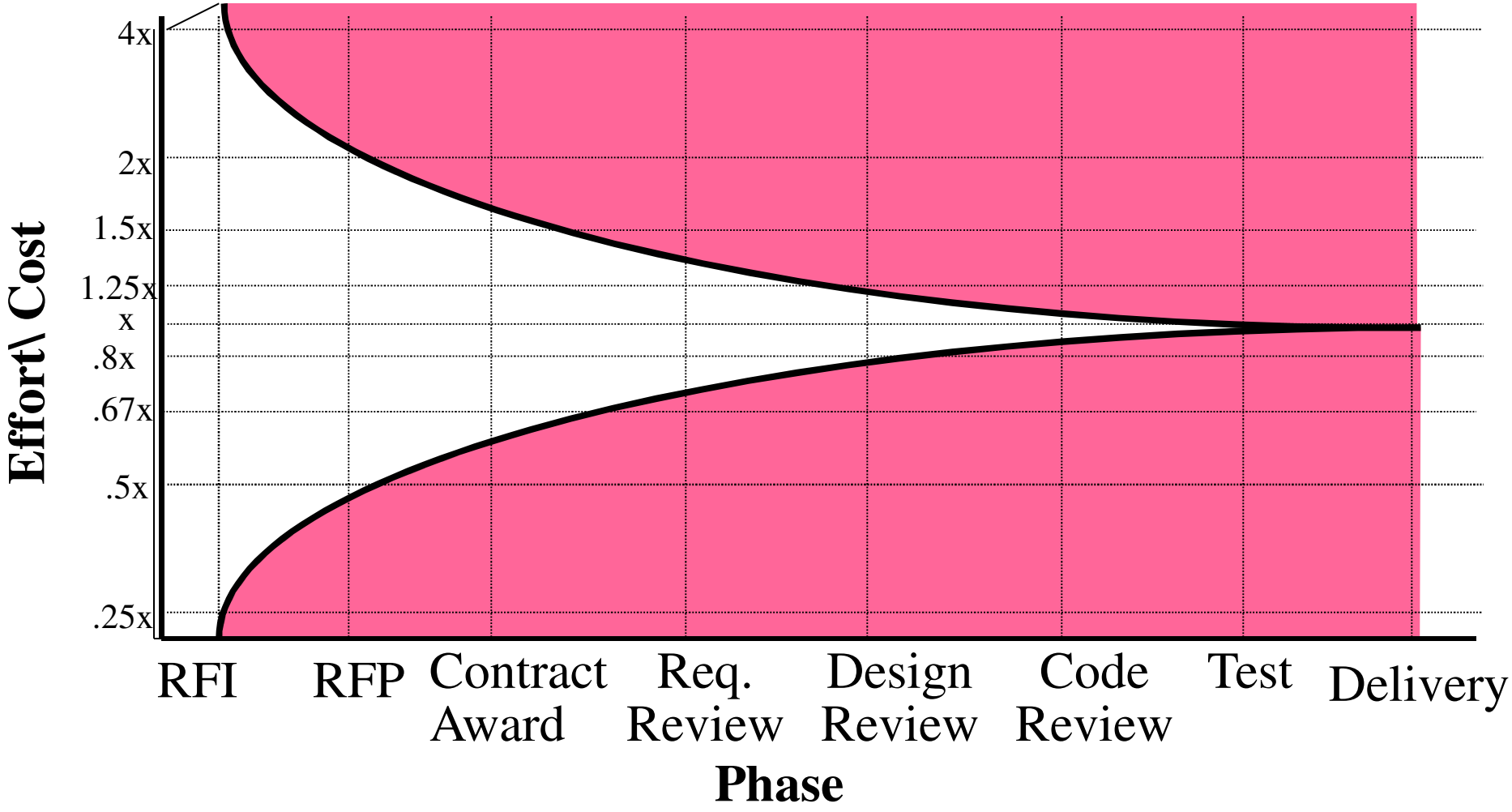


Lifecycles Strengths and Weaknesses

Capability	*Pure Waterfall*	*Code- and - Fix*	*Spiral*	*Modified Waterfall*	*Prototype*
Poorly understood requirements	Poor	Poor	Excellent	Fair to Excellent	Excellent
Poor Architecture	Poor	Poor	Excellent	Fair to Excellent	Poor to Fair
Highly Reliable System	Excellent	Poor	Excellent	Excellent	Fair
System Growth Built in	Excellent	Poor to Fair	Excellent	Excellent	Excellent
Risk Management	Poor	Poor	Excellent	Fair	Fair
Predefined Schedule	Fair	Poor	Fair	Fair	Poor
Midcourse Correction	Poor	unknown	Fair	Fair	Excellent
Customer Visibility	Poor	Fair	Excellent	Fair	Excellent
Management Visibility	Fair	Poor	Excellent	Fair to Excellent	Fair
Low Management and developer skill level	Fair	Excellent	Poor	Poor to Fair	Poor
Low Overhead	Poor	Excellent	Fair	Excellent	Fair

Rapid Development (McConnell, 96)

Change Possibility

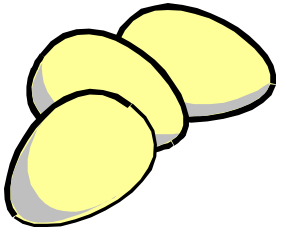


Cost Models for Future Life Cycle Processes: COCOMO 2.0 (Boehm, 1995)

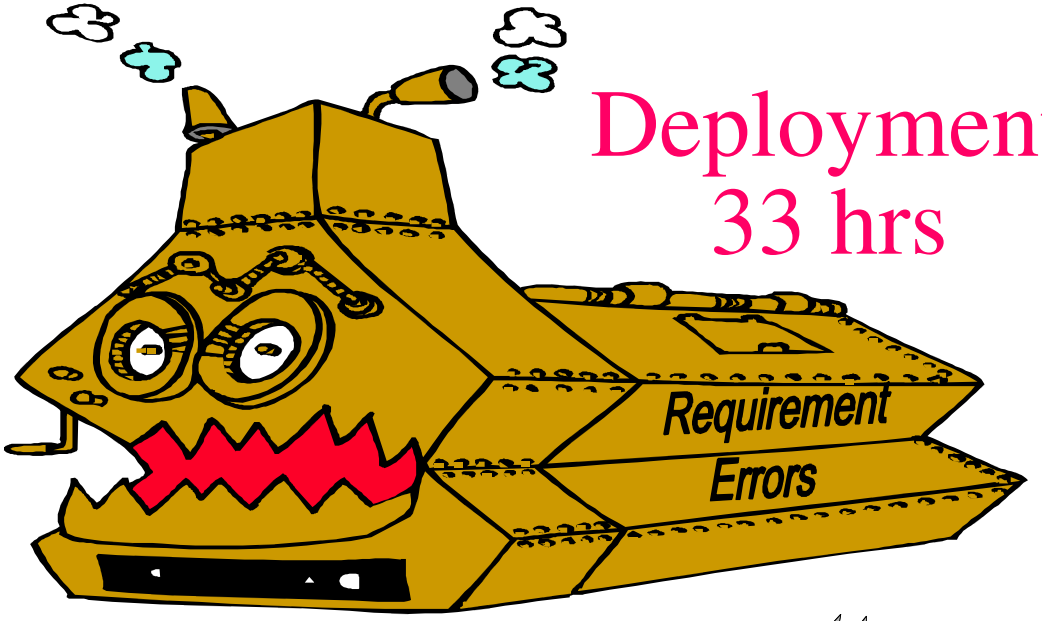
Finding Requirement Errors

U.S. AIR FORCE

S T S C

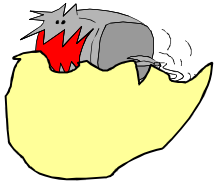


Requirements
Review 1 hr

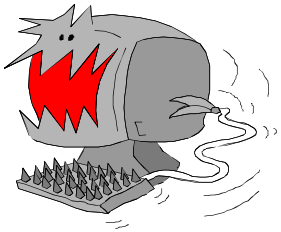


Deployment
33 hrs

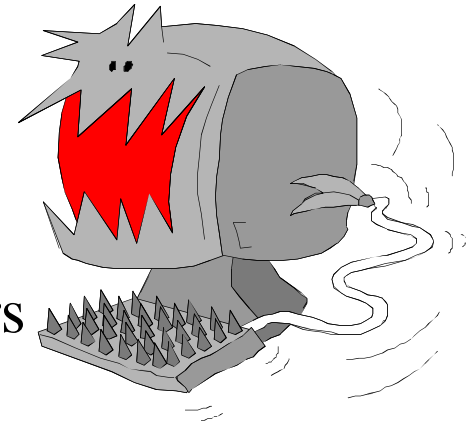
Design
2.5 hrs



Code
13 hrs

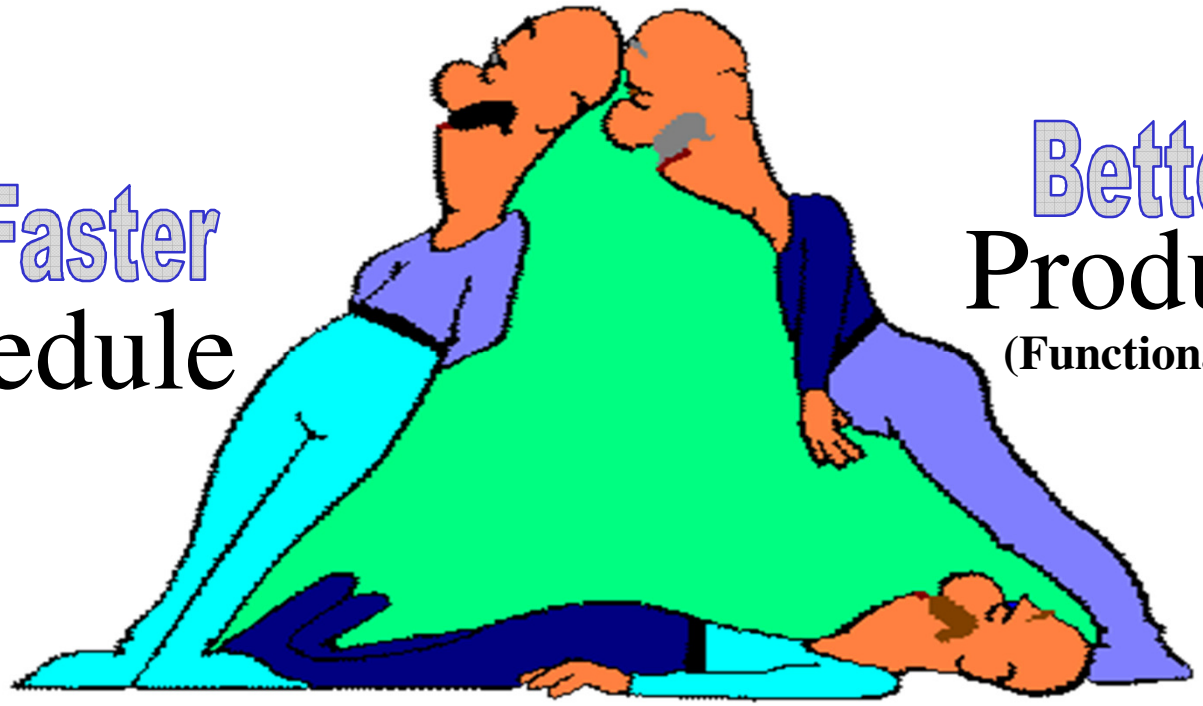


Test
17 hrs



The Development Triangle

You can control change to only two sides of a triangle; The third side must freely adapt – or else it's not a triangle anymore.



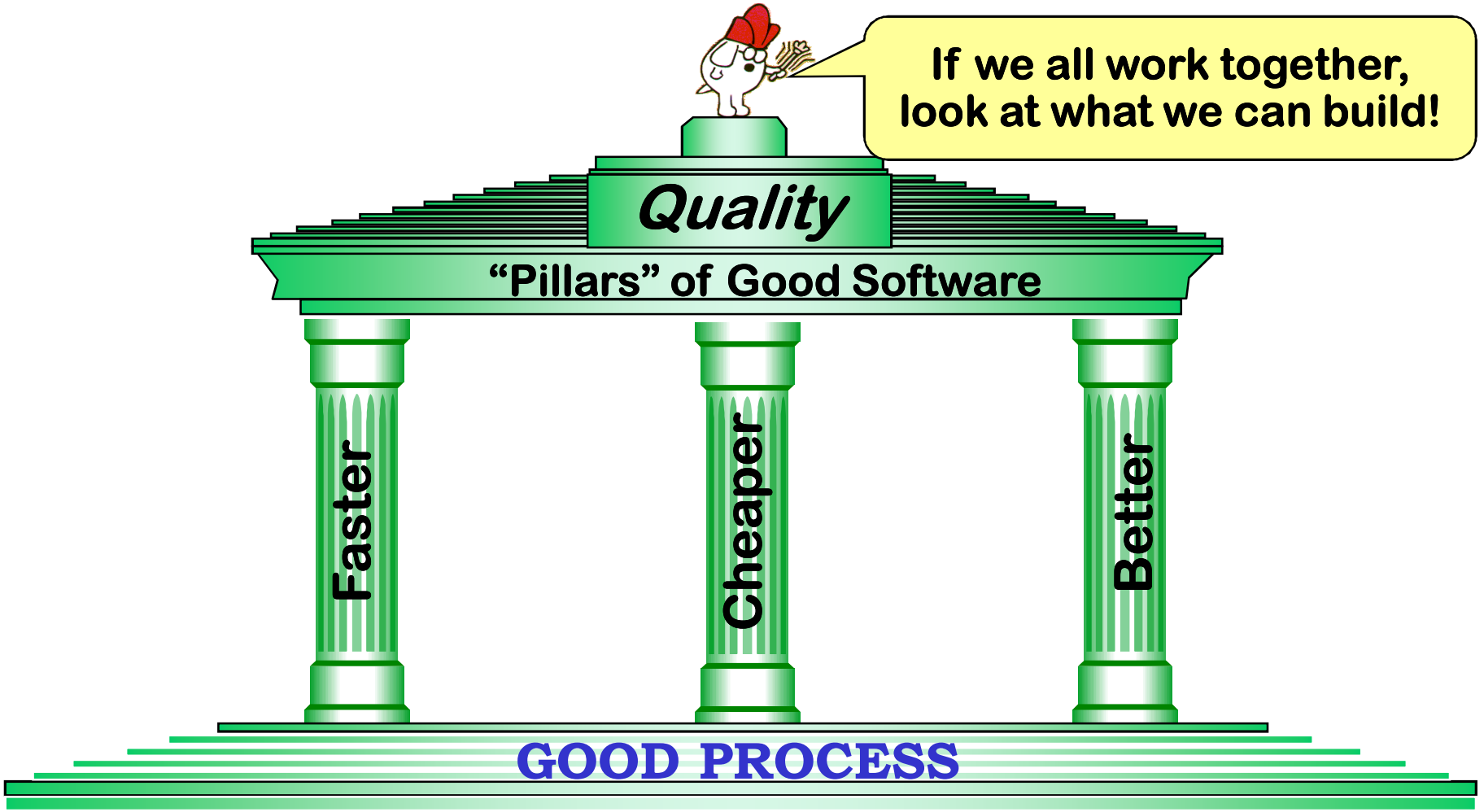
Faster
Schedule

Better
Product
(Functionality + Quality)

Cheaper Cost

Software is both a source of amusement and engineering achievement.

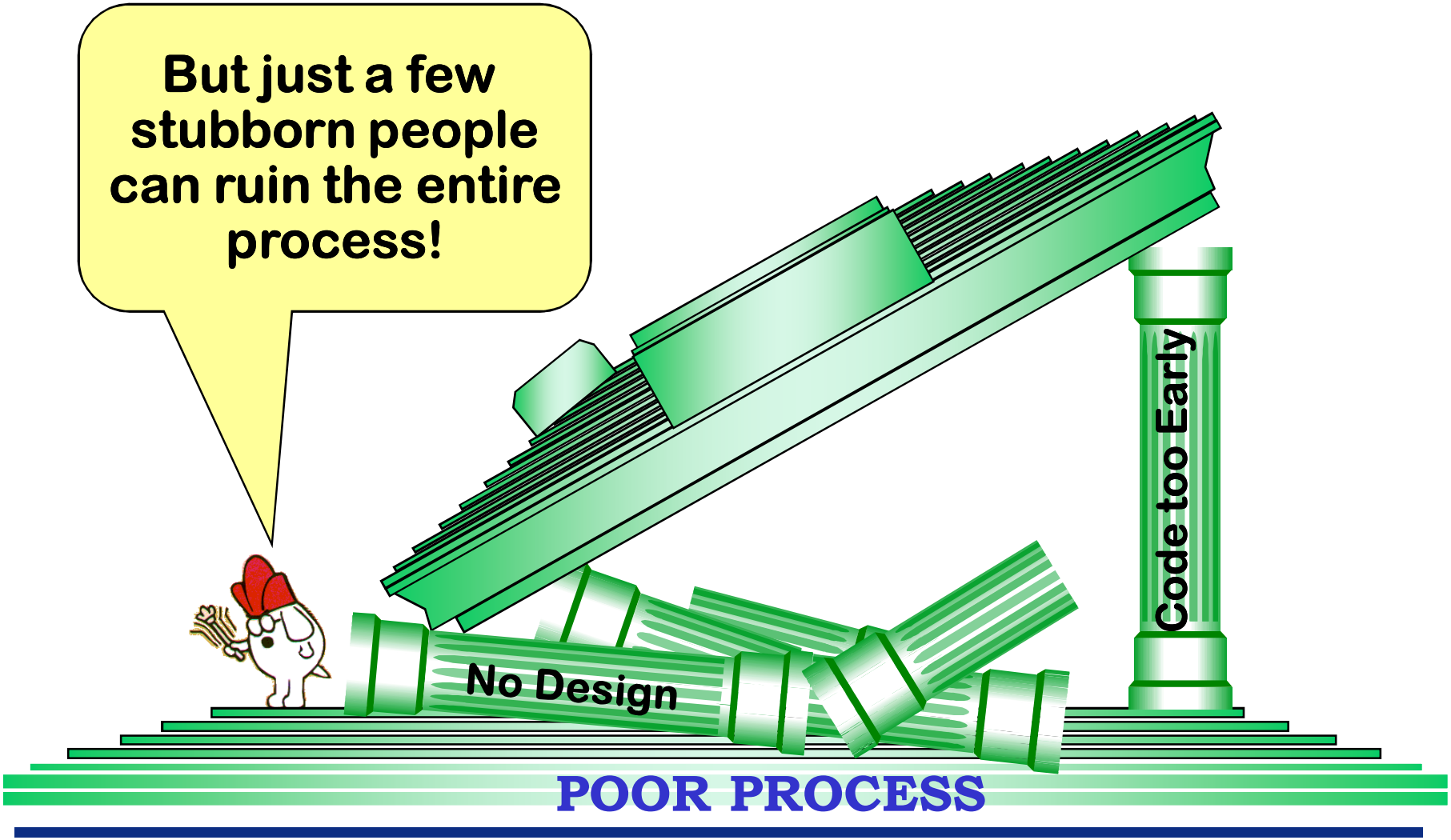
The Temple of Software Engineering



U.S. AIR FORCE

The Temple of Software Engineering

S T S C



More Information

You can go on the internet and search for more information on the topics discussed and /or check on the links below:

*www.**dau**.mil*

*www.**nps**.edu*

*www.**afit**.edu/about.cfm*

http://www.usability.gov/templates/docs/u-test_plan_template.doc

*www.**uml**.org*